Binaria dla systemu CROOK

Jan Rudzinski
Warszawa, listopad '85

W systemie binariow wystepuja zbiory modulow relokowalnych i absolutnych. Zbiory modulow relokowalnych moga zawierac jeden lub wiecej takich modulow, przy czym na zbior moze byc ewentualnie nalozona struktura biblioteki, wystepuja wiec zbiory sekwencyjne i biblioteczne.

Obsluga modulow binarnych zajmuja sie w systemie nastepujace programy:
asembler GASS - generuje zbiory sekwencyjne, zawierajace jeden lub wiecej modulow relokowalnych,
link-editor LINK - laczy moduly relokowalne, tworzac z kilku modulow jeden, lub zamienia moduly relokowalne na jeden zbior binariow absolutnych, ktory moze zawierac wiele nakladek danego programu,
bibliotekarz LIAR - tworzy zbiory biblioteczne i pozwala dolaczac do nich moduly badz je usuwac.

Zbiory binariow absolutnych sa wykorzystywane przez ekstrakody systemowe RUN, ktory kreuje zadanie, laduje nakladke poziomu zerowego ze wskazanego zbioru i uruchamia ja od adresu transferowego, oraz OVERLAY ktory dolacza nakladke o podanej nazwie znajdujaca sie w zbiorze z ktorego zaladowano nakladke glowna.

Mozliwe jest dolaczanie dodatkowych programow uzytkownika, ktore moga wykonywac zadane operacje na modulach relokowalnych, badz je kreowac. W dodatkach podane beda szczegolowe wymagania na strukture zbioru modulow.

Zbiory modulow o strukturze bibliotecznej roznia sie od zbiorow sekwencyjnych slownikiem dolaczonym na koncu zbioru, tzn. za ostatnim modulem wchodzacym w sklad biblioteki. Struktura zbiorow jest tak dobrana, ze we wszystkich kontekstach, w ktorych na wejsciu programu wymagany jest zbior sekwencyjny, moze rowniez wystapic biblioteka; istnieja natomiast sytuacje, w ktorych niezbedne jest istnienie na zbiorze struktury bibliotecznej, a zwykle zbiory sekwencyjne nie moga byc zastosowane.

Moduly relokowalne charakteryzuja sie nastepujacymi, widzialnymi przez programy je obslugujace atrybutami:

nazwa modulu,
adres transferowy: wyrozniony adres, pod ktory nalezy przekazac sterowanie po zaladowaniu modulu; moze w module nie wystapic,
symbole eksportowane: (w skrocie eksporty) nazwy adresow wewnatrz modulu, ktore maja byc znane w innych modulach
symbole importowane: (w skrocie importy) nazwy adresow obiektow, ktore maja byc widziane w danym module, ale w nim nie wystepuja,
zmienne zewnetrzne: omowione beda pozniej.

Warto wiedziec, ze nazwy modulow nie maja nic wspolnego z nazwami obiektow zewnetrznych danego modulu, w szczegolnosci moga sie z takimi nazwami pokrywac bez zadnych konsekwencji dla sposbu wykorzystania modulu.

Kazdy eksport, import i zmienna zewnetrzna ma okreslony typ relokacji, a mianowicie Byte, Code lub Data, przy czym symbole z obszaru Byte wyrazone sa z bajtach, natomiast z obszarow Code i Data w slowach. Typ obszaru nie jest rozszerzeniem nazwy obiektu w tym sensie, ze obecnosc dwu obiektow o tej samej nazwie ale innym typie relokacji sygnalizowana jest jako blad.

Asembler GASS generuje po jednym module dla kazdej jednostki asemblacji, ograniczonej dyrektywami:

```
        .unit     xxx
        .end      [yyy]
```

przy czym nazwa modulu pokrywa sie z nazwa podana w dyrektywie .unit.
Eksporty, importy i zmienne zewnetrzne generowane sa zgodnie z dyrektywami

```
        .exp      lista symboli
        .imp      lista symboli
nazwa   .xvar     dlugosc
```

natomiast adres transferowy jest adresem eksportu, ktorego nazwe podano po dyrektywie .end.
Asembler GASS wpisuje rowniez w tabliczce komentarza czas i date utworzenia modulu, swoj identyfikator (GASSnumer-wersji) i ewentualny komentarz uzytkownika (dyrektywa .rem)

Bibliotekarz LIAR zamienia sekwencyjny zbior modulow na biblioteke oraz dodaje do biblioteki wskazane moduly ze wskazanych zbiorow sekwencyjnych (lub bibliotek), a takze je usuwa. Nie zmienia zadnych atrybutow modulow, a jedynym atrybutem istotnym dla niego jest nazwa modulu: w wypadku dolaczania modulu o nazwie pokrywajacej sie z nazwa modulu juz wystepujacego w bibliotece, ten ostatni jest zaslaniany. Moduly zasloniete nie sa widoczne dla zadnych programow wykorzystujacych biblioteke, ale moga byc odsloniete po usunieciu z biblioteki modulu ktory je zaslonil. Ilosc kolejnych wersji modulu o tej samej nazwie nie jest ograniczona, a ich organizacja przypomina stos.
Wazna cecha bibliotek jest to, ze przy ich sekwencyjnym przetwarzaniu kolejnosc napotykania poszczegolnych modulow jest nieokreslona, zaleca sie wiec, aby programy laczone z wielu modulow zachowywaly sie niezmienniczo ze wzgledu na kolejnosc umieszczenia modulow w pamieci. W systemie istnieja metody, pozwalajace na precyzyjne lokowanie zadanych fragmentow kodu we wlasciwe miejsca pamieci.

Link-editor LINK moze pracowac w dwoch trybach: relokowalnym i absolutnym. W trybie relokowalnym wynikiem pracy jest jeden modul relokowalny, ktory moze byc umieszczony w bibliotece lub laczony z innymi modulami w kolejnym przebiegu LINKa, nie moze natymiast byc wykonany.
W trybie absolutnym produkowany jest jeden zbior zawierajacy binaria absolutne, byc moze podzielone na rozne nakladki. Zbior ten moze byc zaladowany i wykonany, natomiast nie mozna go dalej przetwarzac tu opisanymi programami.
Zbiory wejsciowe do LINKa mozna podzielic na zbiory zwykle i biblioteki. Zbiory zwykle moga byc zbiorami sekwencyjnymi lub bibliotecznymi, natomiast biblioteki musza miec strukture biblioteczna. Ze zbiorow zwyklych trafiaja do modulu wynikowego wszystkie, badz wybrane explicite przez uzytkownika moduly, natomiast biblioteki przeszukiwane sa automatycznie i wybierane z nich sa te i tylko te moduly, ktore zawieraja eksporty, odpowiadajace nie zwiazanym jeszcze importom.
Omowimy najpierw zachowanie sie LINKa w trybie relokowalnym.

Adres transferowy moze wystapic w co najwyzej jednym module wejsciowym i jesli taki wystepuje, przekazywany jest do modulu wynikowego.
Nazwa modulu wynikowego jest nazwa tego modulu wejsciowego, ktory zawieral adres transferowy, a jesli takiego nie ma, pierwszego napotkanego.
W komentarzu do modulu LINK podaje czas i date polaczenia modulu i swoj identyfikator (LINK numer wersji), gubi natomiast ewentualne komentarze uzytkownika.
Nazwy importow sa uzgadniane z eksportami, przy czym nazwy

nie uzgodnione przechodza do modulu wynikowego. Wsrod wszystkich modulow wejsciowych eksport o danej nazwie powinien wystapic tylko raz.
W roznych modulach wejsciowych moga wystapic zmienne zewnetrzne o tej samej nazwie, ktorym w module wynikowym odpowiadac bedzie jedna zmienna zewnetrzna o tej nazwie i o dlugosci najwiekszej z nich. Obiekt odpowiadajacy tej zmiennej powstanie poza cialem modulu wynikowego.
W wypadku wystapienia eksportu o tej samej nazwie co zmienna zewnetrzna, jest ona utozsamiana z tym eksportem (wchlaniana prze niego) i w module wynikowym juz nie wystepuje. Zmienna zewnetrzna o wyroznionej nazwie "$" nie moze zostac wchlonieta przez zaden eksport. Modul moze importowac nazwe zmiennej zewnetrznej.

Zarowno para eksport/import jak i zmienne zewnetrzne umozliwiaja dzielenie obszaru pamieci przez niezaleznie asemblowane moduly, jednak przy eksporcie wystepuje wyrazna asymetria: symbol eksportowany jest przez dokladnie jeden modul, w ktorym zarezerwowno odpowiedni obszar, a pozostale moduly ten symbol importuja, mamy wiec asymetrie sytuacji nadawcy i odbiorcy. Przy zmiennych zewnetrznych zachowana jest symetria: dowolna ilosc modulow moze deklarowac w identyczny sposob zmienna zewnetrzna o tej samej nazwie, ktora w rezultacie usytuowana bedzie przez system poza obszarem tych modulow. Przewiduje sie generowanie zmiennych zewnetrznych przez kompilator jezyka C (odpowiednik pojecia zmiennej zewnetrznej w tym jezyku) i przez kompilator fortranu (odpowiednik bloku COMMON z nazwa).

Przy pracy w trybie absolutnym LINKer wykonuje w sposob wyzejPrzy pracy w trybie absolutnym LINKer wykonuje w sposob
wyzej opisany sklejenie modulow tworzacych poszczegolne nakladki, a nastepnie ustala wzajemne polozenie nakladek, oblicza polozenie zmiennych zewnetrznych oraz generuje zbior zawierajacy binaria absolutne utworzonych nakladek. Wystepuje dodatkowy warunek, a mianowicie nakladka glowna - poziomu zerowego musi miec adres transferowy; pozostale nakladki taki adres miec moga.
Wewnatrz pliku wynikowego kazda nakladka winna miec unikalna nazwe, po ktorej rozpozna ja ekstrakod OVERLAY. Nazwa nakladki glownej tworzona jest w mysl tych samych regul, co nazwa modulu wynikowego przy pracy w trybie relokowalnym, natomiast nazwy pozostalych nakladek moga byc zadane dyrektywami LINKa, a w ich braku ustalane sa rowniez w mysl tych samych regul.

Wszystkie nazwy importow i eksportow traktowane sa globalnie, tzn. w dowolnej nakladce mozna dysponowac adresem symbolu eksportowanego z dowolnej innej nakladki, mozliwe jest wiec laczenie symboli "w dol" (procedura w nakladce

korzysta z procedur swojego przodka), "w gore" (proces wywoluje procedure w zaladowanej nakladce) i "w bok" (adres obiektu z sasiedniej nakladki przekazywany jest w dol jako parametr).
W zwiazku z wyzej powiedzianym, dowolny modul przylaczony automatycznie z biblioteki pojawi sie co najwyzej raz. Jesli wiec w dwu roznych nakladkach ma byc wykorzystana ta sama procedura biblioteczna, nalezy ja dolaczyc we wspolnym przodku tych nakladek. Najwygodniej jest zapewne dolaczyc wszystkie procedury biblioteczne juz w nakladce poziomu zerowego.
W razie potrzeby mozna przygotowac nakladke wraz z modulem bibliotecznym osobnym przebiegiem LINKa w trybie relokowalnym, a nastepnie dolaczyc te nakladke z juz wbudowanym modulem bibliotecznym we wlasciwe miejsce struktury.

Istnieja pewne dodatkowe warunki na zmienne zewnetrzne w programach nakladkowanych. Zalozmy, ze zmienne zewnetrzne o tej samej nazwie name wystapily w nakladkach Ox1, Ox2,... Oxm. Niech D(name) = D(Ox1, Ox2,...Oxm) bedzie minimalnym poddrzewem zawierajacym te nakladki. Wtedy w korzeniu drzewa D(name) musi wystapic badz zmienna zewnetrzna o tej nazwie, badz wchlaniajacy ja eksport. Jesli w korzeniu wystepuje zmienna musi ona dodatkowo miec dlugosc niemniejsza, niz dlugosc wszystkich innych zmiennych zewnetrznych o tej nazwie.
Powyzsze ograniczenie nie dotyczy zmiennej zewnetrznej o wyroznionej nazwie "$", ktora moze wystapic w dowolnych nakladkach, natomiast nie moze byc wchlonieta przez eksport. Zmienna $ lokowana jest w wolnym obszarze pamieci, tak, aby mozna w niej bylo umiescic rozszeralne struktury danych. W kompilatorze fortranu zmienna ta bedzie odpowiadac blokowi COMMON bez nazwy.

Przy rozmieszczaniu binariow absolutnych w pamieci LINKer stosuje sie do nastepujacych zasad:
Slowo o numerze zero jest zawsze wolne, nastepnie umieszczana jest zawartosc sekcji BYTE wszystkich nakladek, nastepnie zawartosc sekcji CODE i na koncu DATA. Oznacza to, ze sekcje B C i D nie beda nigdy przeplecione, natomiast przeplatanie moze dotyczyc sekcji roznych nakladek. Obszar wolny, nie zajety przez zadne sekcje, przeznaczony jest na zmienna zewnetrzna "$", ktora ma pelnic funkcje zmiennej rozszerzalnej. Mozliwe sa wiec nastepujace rozmieszczenia sekcji programu wynikowego, zaleznie od typu zmiennej zewnetrznej $:
[0] Byte Code Data (nie ma)
[0] Byte Code Data $-xvar [koniec pamieci] (typ data)
[0] Byte Code $-xvar Data [koniec pamieci] (typ code)
[0] Byte $-xvar Code Data [koniec pamieci] (typ byte)

Rozwazmy przyklad. Niech nakladka poziomu zero sklada sie z dwu modulow: A0 i B0, zawierajacych tylko sekcje Code i Data, a takze zmienna zewnetrzna X i zmienna $ (typu data). Z kolei nakladki poziomu pierwszego to moduly A1a i A1b, przy czym A1a zawiera sekcje Byte i Data, natomiast A1b sekcje Code i Byte oraz zmienna zewnetrzna XX typu Data; zalozmy przy tym, ze zmienna XX jest krotsza od obszaru Data modulu A1a, natomiast obszar Byte dluzszy jest w module A1b. Po zaladowaniu programu rozmieszczenie elementow w pamieci bedzie jak nastepuje (gwiazdki oznaczaja obszar niewykorzystany):

[0] ******** A0(C) B0(C) ****** A0(D) B0(D) X ****** $

Po zaladowaniu nakladki A1a uzyskamy:

[0] A1a(B) * A0(C) B0(C) ****** A0(D) B0(D) X A1a(D) $

a po zaladowaniu A1b:

[0] A1b(B) A0(C) B0(C) A1b(C) A0(D) B0(D) X XX * $

Obszar niewykorzystany miedzy zmienna XX a $ zawiera obecnie "resztki" obszaru Data nakladki A1a.

Binaria absolutne generowane przez LINKa ladowane sa do pamieci i uruchomiane przez ekstrakody RUN i OVERLAY (dokladna postac ich wywolania w Dodatku G).
Parametrem ekstrakodu RUN jest nazwa pliku, zawierajacego binaria absolutne, argumenty ktore maja byc przekazane do programu (sposob ich przekazania zalezy od systemu) i ewentualnie inne parametry, niezbedne dla prawidlowego uruchomienia programu przez system.
Efektem pracy ekstrakodu jest uruchomienie programu, dzialajacego w przypisanym mu obszarze pamieci. Wszystkie pozycje pamieci okreslone w tablicach binariow absolutnych sa wlasciwie ustawione, natomiast zawartosc pozostalych slow jest nieokreslona.
Ladowana jest zawsze nakladka glowna (poziomu zerowego), a jej wykonania rozpoczyna sie od adresu transferowego. Program tak zaladowany dysponuje wektorem argumentow, ktory moze odczytac w sposob zalezny od systemu. Ekstrakod RUN bedzie najczesciej wywolywany przez jezyk sterowania praca, np. OSL.

Ekstrakod OVERLAY powinien byc wywolywany przez wykonujacy sie program. Jego parametrem jest nazwa nakladki, ktora bedzie poszukiwana w tym samym pliku, z ktorego zaladowano program glowny.
W efekcie pozytywnego wykonania ekstrakodu do pamieci operacyjnej wprowadzi sie zawartosc tabliczek opisujacych

nakladke o zadanej nazwie. Zawartosc zadnych innych slow pamieci nie ulegnie zmianie. Jesli ladowana nakladka ma adres transferowy, nastapi skok pod ten adres po wyjsciu z ekstrakodu, natomiast przy jego braku ekstrakod zwroci sterowanie w miejsce, z ktorego byl wywolany.

Ekstrakody RUN i OVERLAY przydzielaja zadaniu te i tylko te ramki pamieci, ktore niezbedne sa do pomieszczenia obszarow B,C i D danej nakladki. Zmienne zewnetrzne oraz obszary rezerwowane przez dyrektywe .res nie sa inicjowane, ale przydzielana jest dla nich ramka pamieci. Dla zmiennej $ rezerwuje sie tylko te ramki pamieci, ktore sa niezbedne dla zmieszczenie jej w zadeklarowanej dlugosci.
Przydzial i zwalnianie ramek pamieci (po 4K) po zaladowaniu nakladki zapewniaja ekstrakody TAKEMEM i GIVEMEM, ktorych argumentem jest opis fragmentu pamieci ktorego ma dotyczyc operacja. Uzycie ekstrakodu TAKEMEM do obszaru z pamiecia juz przydzielona i GIVEMEM do obszaru bez pamieci jest operacja pusta.

Wywolanie przez program ekstrakodu RUN daje efekt lancuchowania programu, ale bez mozliwosci komunikacji miedzy kolejno wolanymi programami. Najwlasciwszym rozwiazaniem dla lancuchowania programow z zachowaniem w pamieci wynikow posrednich jest dolaczenie do poszczegolnych przebiegow nakladki poziomu zeru, ktorej jedyna rola jest wywolanie pierwszego przebiegu i zadeklarowanie zmiennych zewnetrznych. W takiej sytucji kolejno wolajace sie przebiegi beda mogly komunikowac sie przez zmienne zewnetrzne poziomu zero i przez zmienna $. Jezeli kolejne przebiegi korzystaja z procedur bibliotecznych, wskazane jest dolaczyc i te procedury do nakladki bazowej, ktorej tresc moze byc jak nastepuje:

```
 .unit starter
 .use code
 .imp sin, writedouble,
; import procedur bibliotecznych
 .use data
table .xvar 400 ; tablica wspolna dla przebiegow
 .use code
 .exp starter
starter ; adres transferowy
; ustaw parametry wywolania nakladki PASS1
 ptr OVERLAY
 .end starter
```

Jesli kolejno wolajace sie przebiegi beda wyposazone w adresy transferowe, nie zajdzie potrzeba wracania do modulu starter przy zmianie przebiegu.

Dodatek A. Struktura zbiorow binarnych jako plikow

Zbiory binariow relokowalnych sekwencyjne i biblioteczne zawieraja ciag umieszczonych jeden za drugim modulow. W zbiorach sekwencyjnych za ostatnim modulem wystepuje koniec pliku, natomiast w zbiorach bibliotecznych umieszcza sie tam kartoteke. Pierwsze slowo kartoteki sluzy do jej latwego zidentyfikowania: zawiera na lewym bajcie znak: x'C0'. Program przetwarzajacy plik biblioteczny sekwencyjnie winien traktowac slowo tej postaci napotkane za koncem modulu jako koniec pliku.
Pusta biblioteka zaczyna sie od razu od kartoteki, nie zawierajac zadnego modulu.

W systemie CROOK zbiory binariow relokowalnych sekwencyjne maja typ OBJ, zbiory bibliteczne typ LIB, a zbiory binariow absolutnych sa typu BIN. Dla odroznionych binariow absolutnym od zbiorow binarneych dawnego typu przewidziany bedzie stosowny bit w tabliczce opisu pliku. (parametr ATR jest zgodny ze standardem systemu CROK-4 i ustawiany jest wstepneie na o'60').
Parametr LEN wyzej wymienionych typow zbiorow zawiera jak zwykle dlugosc zbioru mierzona w blokach, parametr PAR1 = LEN -1, a parametr PAR2 zawiera offset ostatniego slowa w ostatnim bloku, tzn. PAR2 = 0 oznacza, ze istotne jest jedno slowo ostatniego bloku, a PAR2 = 255 znaczy, ze ostatni blok jest pelny.

Dodatek B. Struktura modulow relokowalnych

Pojedynczy modul relokowalny sklada sie z nastepujacych czesci:

- specyfikacja
- komentarz
- tablica symboli eksportowanych
- tablica symboli importowanych
- tablica zmiennych zewnetrznych
- kod

Wszystkie liczby w module relokowalnym traktowane sa jako adresy, tzn. sa z zakresu 0..65535.

Dwubitowe pole R okreslaja typ sekcji badz typ relokacji wartosci interpretowane jest nastepujaco:

R = 0	A	(wartosc absolutna)
R = 1	B	(relokacja typu Byte)
R = 2	C	(relokacja typu Code)
R = 3	D	(relokacja typu Data)

S p e c y f i k a c j a

Specyfikacja modulu relokowalnego zajmuje 21 slow.

slowo 0 wyroznik typu modulu.
Wartosc 17 na lewym bajcie oznacza modul relokowalny taki, jak opisany w niniejszym dokumencie.

slowo 1 suma kontrolna
Suma kontrolna pozwala sprawdzic spojnosc specyfikacji. W poprawnej specyfikacji operacja XOR (lub wylaczne) zastosowane do wszystkich slow winna wyniesc zero.

slowa 2-5 nazwa modulu.
Nazwa modulu moze zawierac duze i male litery, znaki $ (dolar), Nazwy krotsze od 8 znakow dopelniane sa spacjami z prawej strony.

slowo 6 atrybuty.

	W	r	r	E	I	f	f	r	
bity	0	1	2	3	4	5	6	7	8-15

Pola maja nastepujace znaczenie:
W - (warning) w module wystepuja bledy, ale zachowuje on spojnosc,
E - (entry) adres transferowy (punkt wejscia) dla modulu jest okreslony i opisany w slowie 9 specyfikacji,
I - (invisible) modul jest spojny, ale winien byc ignorowany przez wiekszosc programow systemowych. Bit I ustawiany jest i gaszony przez program LIAR.
f - (free) zarezerwowane dla przyszlych rozszerzen.
r - zarezerwowane dla programu LIAR

slowa 7-8 adres nastepnego modulu. W slowach tych podany jest adres (w mierzona w slowach, zapisany jako dluga liczba calkowita) poczatku specyfikacji nastepnego, byc moze niewidzialnego, modulu. W ostatnim module podany jest adres poczatku kartoteki lub gdy jej nie ma, adres pierwszego slowa za koncem pliku. Slowa 7-8 moga byc zerowe, oznacza to, ze adres nastepnego modulu nie jest znany.

slowo 9 adres transferowy.
Index w tablicy eksportow symbolu opisujacego adres transferowy.

slowa 10-11 zarezerwowane dla programu LIAR.

slowa 12-14 zarezerwowane dla przyszlych rozszerzen.

slowo 15 dlugosc sekcji Byte.

slowo 16 dlugosc sekcji Code.

slowo 17 dlugosc sekcji Data.

Dlugosc wszystkich sekcji podana jest w slowach.

slowo 18 calkowita dlugosc sekcji Byte.
slowo 19 calkowita dlugosc sekcji Code.

slowo 20 calkowita dlugosc sekcji Data.

Calkowita dlugosc sekcji jest suma dlugosci sekcji i dlugosci wszystkich zmiennych zewnetrznych zadeklarowanych w tej sekcji przez modul.

Dodatek C. Struktura kartotek modulow bibliotecznych

W zbiorach bibliotecznych wykorzystywane sa niektore slowa specyfikacji modulu binarnego.
W slowie o numerze 6 (atrybuty modulu) LIAR wykorzystuje nastepujace bity:

bit 1 - modul posiada poprzednia wersje, ktorej adres zawarty jest w slowach 10-11 specyfikacji,
bit 2 - modul jest definitywnie usuniety dyrektywa D,
bit 4 - modul jest niewidzialny (usuniety lub zasloniety),
bit 7 - moze byc ustawiony tylko w pierwszym module zbioru, sygnalizuje, ze zbior ma strukture biblioteczna; adres kartoteki znajduje sie wtedy w slowach 10-11.

Slowa 10-11 specyfikacji zawieraja adres kartoteki jesli specyfikacja opisuje pierwszy fizycznie modul zbioru (byc moze niewidzialny), badz adres poprzedniej wersji danego modulu, o ile jest ustawiony bit 1 w slowie atrybutow. Adres podany jest jako dluga liczba calkowita i wyrazony jest w slowach liczac od poczatku zbioru.

Kartoteka sklada sie z naglowka liczacego trzy slowa i z ramek opisujacych moduly i ich eksporty po 7 slow kazda.
Slowo 0 naglowka zawiera wyroznik kartoteki, tzn. wartosc x'C0' na lewym bajcie, slowo 1 opisuje dlugosc kartoteki (liczona lacznie z naglowkiem), a slowo 2 jest suma kontrolna i zawiera wynik operacji XOR na slowach 0 i 1.

Ramka opisu modulu badz eksportu zawiera w slowach 0-3 nazwe obiektu, w slowie 4 atrybuty obiektu i w slowach 5-6 adres poczatku modulu opisanego przez ramke badz zawierajacego dany eksport. Bity slowa atrybutow sa nastepujace:

bit 14 - ramka opisuje modul (=1) badz eksport (=0),
bit 15 - zawsze 1

a opisuje modul (=1) badz eksport (=0),
bit 15 - zawsze 1

Dodatek D. Struktura zbiorow binariow absolutnych

Zbior binariow absolutnych sklada sie z co najmniej jednego bloku opisu nakladek, z opcjonalnego bloku zadan systemowych oraz z tresci binariow. Opisy nakladek mieszcza sie w bloku o numerze 0, a ewentualne nastepne bloki sa powiazane w liste. Blok o numerze zero ma nastepujaca strukture:
Slowo o numerze zero - na lewym bajcie wyroznik binariow absolutnych, tzn. x'AB', a na bitach 9..15 prawego bajtu ilosc nakladek opisanych w tym bloku. Bit 8 tego slowa jest zapalony, jesli dany blok nie jest ostatni na liscie.
Slowo o numerze jeden - adres liczony w blokach bloku zadan systemowych lub zero, jesli nie wystepuje.
Dalej wystepuja tabliczki opisu nakladki o dlugosci szesciu slow (do 42 w jednym bloku). Za ostatnia tabliczka w ogole powinny wystapic dwa slowa zawierajace zero, natomiast za ostatnia tabliczka w danym bloku nalezy umiescic slowo zawierajace zero i po nim adres liczony w blokach nastepnego bloku na liscie.
Nakladka glowna, ladowana przez ekstrakod RUN, opisana jest w pierwszej tabliczce pierwszego bloku (tzn. bloku numer 0).

Dodatkowe bloki opisu nakladek maja strukture identyczna, jak pierwszy, z tym, ze zawartosc slowa o numerze jeden winna byc ignorowana.

Tabliczka opisu nakladki w slowach 0-3 zawiera nazwe nakladki po ktorej bedzie ona rozpoznana przez ekstrakod OVERLAY, (reguly tworzenia nazwy sa identyczne, jak przy nazwach madulow), a w slowach 4-5 adres poczatku binariow danej nakladki liczony w slowach od poczatku pliku.

Struktura bloku zadan dodatkowych zostanie ustalona pozniej. Moze on zawierac informacje, ktore ulatwia zarzadzanie zasobami maszyny, np. maksymalna ilosc ramek pamieci, ktorej zazada proces. Przewiduje sie, ze zadania dodatkowe beda ustawiane przez specjalny program.

Opis binariow zawiera w pierwszym slowa wzorzec ladowania pamieci, gdzie kazdy bit oznacza ramke pamieci ktora niezbedna jest dla danej nakladki. Bit 15 (najmniej znaczaczy) oznacza adresy od 0 do x'0FFF', bit 14 adresy x'1000' do x'1FFF' itd.
Drugie slowo opisu binariow zawiera adres transferowy nakladki lub zero, jesli go nie ma.
Dalej wystepuja tabelki trzech roznych rodzajow.
Tabelka ladowania: pierwsze slowo zawiera liczbe z zakresu 1..32755 ktora okresla ilosc slow do zaladowania, drugie slowo zawiera adres, a dalej wystepuje ciag slow, ktore

nalezy umiescic w pamici poczawszy od podanego adresu.
Tabelka inicjacji: pierwsze slowo zawiera liczbe z zakresu -1..-32756, ktora po zanegowaniu oznacza ilosc slow do inicjacji, drugie slowo zawiera adres, a trzecie wartosc ktora nalezy powielic w pamieci od wskazanego adresu.
Tabelka koncowa: pierwsze i jedyne slowo zawiera zero, oznacza ze ladowanie nakladki juz jest zakonczone.

Opis binariow danej nakladki moze zaczynac sie od dowolnego slowa pliku (niekoniecznie na granicy bloku), a kolejnosc obszarow pamieci ladowanych kolejnymi tabelkami moze byc dowolna.

Dodatek E. Parametry wywolania asemblera GASS.

Dodatek F. Dyrektywy i parametry wywolania LINKa

W parametrach wywolania programu LINK mozna umiescic tytuly zbiorow i wskazania opcji. Tytuly zbiorow opisuja zbiory wejsciowe i biblioteczne. Istnieja nastepujce grupy opcji:

Opcje opisu bibliotek:

?LIB - wszystkie tytuly zbiorow wystepujace przed napotkaniem tej opcji opisuja zbiory wejsciowe LINKa, natomiast zbiory za ta opcja sa bibliotekami.
?NSL - (not system library) po przeszukaniu wskazanych przez uzytkownika bibliotek przegladana jest biblioteka systemowa o ustalonej nazwie. Ustawienie opcji ?NSL wstrzymuje przegladanie biblioteki systemowej.

Opcje trybu pracy:

?REL - ustawienie tej opcji powoduje generowanie przez LINKer modulu relokowalnego,
?ABS - ta opcja ustawia LINKa w tryb pracy z generowaniem modulu absolutnego.
Jako domyslny przyjety bedzie tryb pracy absolutny.

Opcje wykonywania komend:

?CMD - opcja ta powoduje czytanie przez LINKa komend ze strumienia INT. Przy jej braku LINK automatycznie laczy moduly napotkane w zadanych zbiorach wejsciowych. Podanie wiecej niz jednego zbioru wejsciowego wyklucza uzycie opcji ?COM. ?CMF tytul zbioru - jak wyzej, ale dyrektywy czytane beda z pliku, podanego jako nastepny parametr. Przy tym trybie pracy bledy syntaktyczne dyrektyw sa bledami fatalnymi.

Jesli przy wywolaniu nie bedzie podany ani jeden zbior wejsciowy, a takze nie wystapi zadna z opcji ?CMD lub ?CMF, jako zbior wejsciowy przyjety bedzie domyslnie STR//BI.

Opcje zbioru wyjsciowego:

?OUT tytul zbioru - wynik pracy LINKa wyslany bedzie na zbior dyskowy o podanej nazwie. Domyslnym zbiorem wyjsciowym

jest STR//BO.

Opcje zachowania eksportow:

?EXP - przy pracy w trybie relokowalnym do zbioru wynikowego trafiaja tylko te eksporty, ktorym nie odpowiadal zaden import. Po ustawieniu opcji ?EXP do modulu wynikowego przejda wszystkie eksporty. Opcja jest ignorowana przy pracy w trybie absolutnym.

Opcje listowania:

Domyslnie LINK wysyla jedynie komunikaty o bledach i ostrzezenia na strumien MON. Mozna ustawic jedna z trzech nastepujacych opcji listowania:

?NOL tytul zbioru - komunikaty o bledach i ostrzezenia beda dodatkowo wysylane do podanego zbioru.
?LIS tytul zbioru - do podanego zbioru bedzie wyslany opis wszystkich modulow wejsciowych (oprocz bibliotecznych) i opis modulu wynikowego badz nakladek wynikowych.
?FUL tytul zbioru - jak wyzej, a ponadto wygenerowana zostanie mapa referencji symboli zewnetrznych, podajaca, w ktorych modulach wystapilo odwolanie do danego symbolu.

Tytuly zbiorow wystepujace po opcjach ?OUT, ?CMF i listowania nie sa traktowane jako opisy zbiorow wejsciowych ani bibliotek.

W trybie pracy z dyrektywami mozliwe sa nastepujace dyrektywy:

L['tytul zbioru] [nazwa modulu]*

ta dyrektywa oznacza dolaczenie modulow o wskazanych nazwach (przy ich braku wszystkich modulow) z podanego zbioru. Przy braku nazwy zbioru przyjmuje sie zbior uzyty w poprzedniej dyrektywie L lub, dla pierwszej takiej dyrektywy, zbior podany jako parametr wejsciowy.

On [nazwa nakladki]

Ta dyrektywa dopuszczalna jest jedynie w trybie absolutnym i pozwala na kreowanie drzewa nakladek. Liczba n oznaczajaca poziom nakladki moze byc wieksza o jeden od numeru nakladki biezacej przy tworzeniu kolejnych, wyzszych nakladek, badz mniejsza/rowna od numeru biezacego przy tworzeniu nowych galezi bocznych. Nazwe nakladki mozna opuscic, przyjeta

bedzie wtedy nazwa modulu z adresem transferowym, a w jego braku nazwa pierwszego napotkanego modulu.
LINK w fazie inicjalizacji wykonuje wewnetrzna dyrektywe

O0

natomiast w trakcie pracy takiej dyrektywy nie mozna powtorzyc: moze istniec dokladnie jedna nakladka poziomu zero.

Q

Dyrektywa Quit inicjuje tworzenie zbioru wynikowego, a po jego utworzeniu konczy prace.

Q!

Ta dyrektywa konczy prace bez tworzenia zbioru wynikowego.

Poczet liter w nazwach dyrektyw i tytulach zbiorow jest nieistotny, natomiast w nazwach modulow i nakladek male i duze litery sa rozrozniane.

Separatorem elementow dyrektyw moze byc ciag spacji, badz przecinek, ewentualnie otoczony spacjami.

Dodatek F. Dyrektywy i parametry wywolania LIARa.

Parametrem wywolania LIARa jest jeden lub dwa tytuly zbiorow. Pierwszy tytul opisuje biblioteke, na ktorej beda wykonywane kolejne operacje. Jesli zbior ten nie istnieje, zostanie utworzony i powstanie na nim slownik biblioteki, na razie pustej.
Jesli zbior wskazany jako pierwszy argument jest zbiorem sekwencyjnym, zostanie do niego dolaczona struktura biblioteczna, natomiast jesli zbior jest biblioteka z niespojna struktura biblioteczna, zostanie ona poprawiona. Utrata spojnosci struktury bibliotecznej moze nastapic, jesli w trakcie aktualizacji biblioteki praca programu zostanie przerwana.

Jesli wystepuje drugi parametr, winien on okreslac zbior modulow relokowalnych (sekwencyjny lub biblioteczny). Wszystkie moduly z tego zbioru zostana wtedy dolaczone do biblioteki, po czym program zakonczy prace.

Przy braku drugiego parametru program LIAR zacznie czytac dyrektywy ze strumienia INT, wysylajac ewentualne ostrzezenia i informacje o bledach na strumien MON, a informacje listujace na strumien OUT.

Dyrektywy i tytuly zbiorow moga byc pisane malymi badz duzymi literami, natomiast w nazwach modulow maleymi badz
duzymi literami, natomiast w nazwach modulow male i duze litery sa rozrozniane. Separatorm elementow dyrektywy jest ciag spacji badz przecinek, ewentualnie otoczony spacjami.

LIAR rozpoznaje nastepujace dyrektywy:

A['tytul zbioru] [nazwa modulu]*

dyrektywa powoduje dolaczenie do biblioteki wskazanych (wszystkich przy braku listy nazw) modulow z podanego zbioru, a przy braku nazwy ze zbioru uzytego w poprzedniej dyrektywie A. Nowy modul zaslania ewentualnie istniejacy w bibliotece modul o tej samej nazwie.

D [nazwa modulu]+

dyrektywa usuwa (nieodwracalnie) z biblioteki moduly o podanej nazwie. Musi wystapic co najmniej jedna nazwa modulu i nie moga sie one powtarzac. Usuniecie modulu powoduje odslonicie poprzedniej wersji modulu o tej samej nazwie, o ile taki byl uprzednio zasloniety.

D! [nazwa modulu]+

jak wyzej, z tym ze usunieciu podlegaja wszystkie wersje modulow o podanej nazwie, zaden wiec modul nie zostanie odsloniety.

L

dyrektywa wyprowadza na strumien OUT informacje o wszystkich modulach. Informacje zawieraja nazwe modulu i nazwy jego wszystkich eksportow.

L!

podawane sa informacje o wszystkich wersjach wszystkich modulow. Informacje zawieraja nazwe modulu oraz czas i date jego utworzenia a takze tresc komentarza, o ile taki istnieje. Opisy wersji tego samego modulu sa podawane spojnie w kolejnosci od aktualnego do najdawniej dolaczonego.

Moduly usuwane sa z biblioteki jedynie logicznie, przez ustawienie odpowiednich bitow w naglowku. LIAR dostarcza dwie dyrektywy umozliwiajace kompresje biblioteki.

C

kompresja biblioteki, w wyniku ktorej usuwane sa fizycznie moduly niedostepne, (usuniete logicznie dyrektywa D).

C!

kompresja, ktora usuwa wszystkie poprzednie wersje, pozostawiajac w bibliotece jedynie wersje aktualna wszystkich modulow.

Q

Dyrektywa konczy prace bibliotekarza LIAR.

Dodatek G. Ekstrakody RUN, OVERLAY, TAKEMEM i GIVEMEM.